6　三角形—重心，外心，内心—

6.1　三角形を描くプログラム

6.1.1　新規プログラム保存

フォルダ名	Sankakukei
ユニット名	SankakukeiU.pas
プロジェクト名	SankakukeiP.dpr

6.1.2　メインイメージ

Image をフォームの中に置く。

Align	alClient
Name	ImageMain

6.1.3　フォーム

Name	FormMain
Caption	三角形
WindowState	wsMaximized

OnCreate ハンドラー

プログラムの中で 斜体は既に書かれている部分を示します。それを参考に記入する位置を判断して，立体の部分を追加または変更してください。

```
(******************* イベントハンドラー ********************)
procedure TFormMain.FormCreate(Sender: TObject);
begin
  TenA := Point(400,227);                    // 頂点を決める
  TenB := Point(300,400);
  TenC := Point(500,400);
  Triangle(TenA,TenB,TenC);                  // 三角形を描く
end; (* FormCreate *)
```

6.1.4　private 宣言

TFormMain の private 部に変数とメソッドのヘッダを追加します。

```
  private
    TenA,TenB,TenC : TPoint;              // 三角形の頂点
    procedure Triangle(A,B,C : TPoint);   // 三角形を描く
```

6.1.5 メソッド実現部

```
(******************** フォームのメソッド ********************)
procedure TFormMain.Triangle(A,B,C : TPoint);
          (* 三角形 ABC を描く *)
begin
  with ImageMain.Canvas do
    begin
      MoveTo(A.X,A.Y);
      LineTo(B.X,B.Y);
      LineTo(C.X,C.Y);
      LineTo(A.X,A.Y);
    end;
end; (* Triangle *)
```

6.1.6 実行

正三角形が描かれます。

6.2 頂点 A を移動する

6.2.1 変数

```
 private
    TenA,TenB,TenC : TPoint;              // 三角形の頂点
    MouseDown : Boolean;                  // マウスボタンを押しているか
    procedure Triangle(A,B,C : TPoint);   // 三角形を描く
    procedure ClearGraph(Color : TColor); // 画面消去
```

6.2.2 メソッド

```
(******************** フォームのメソッド ********************)
procedure TFormMain.ClearGraph(Color : TColor);
          (* Image 全体を Color 色に初期化 *)
begin
  with ImageMain.Canvas do
    begin
      Brush.Color := Color;
      FillRect(ClipRect);
    end;
end; (* ClearGraph *)
```

6.2.3 ImageMain の Mouse 関係のイベントハンドラー

```
procedure TFormMain.ImageMainMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
```

```
  ImageMain.Cursor := crCross;                  // カーソルの形を変更
  Mouse.CursorPos := ClientToScreen(TenA);      // カーソルを点 A の上に移動
  MouseDown := True;                            // ボタンダウン中
end; (* ImageMainMouseDown *)

procedure TFormMain.ImageMainMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  ImageMain.Cursor := crDefault;                // カーソルの形を元に戻す
  MouseDown := False;                           // ボタンダウン終了
end; (* ImageMainMouseUp *)

procedure TFormMain.ImageMainMouseMove(Sender: TObject; Shift: TShiftState;
  X, Y: Integer);
begin
  if MouseDown
    then begin
           ClearGraph(clWhite);                 // 画面を消す
           TenA := Point(X,Y);                  // 点 A を変更
           Triangle(TenA,TenB,TenC);            // 三角形を描く
         end;
end; (* ImageMainMouseMove *)
```

6.2.4　実行

ImageMain のどこかをクリックすると，マウスが点 A の上に移動します。右クリックの場合，マウスカーソルの形が＋に変わります[1]。ボタンを押したままマウスを動かすと，どうなりますか。

6.3　内部を塗る

ブラシで塗るのは，FloodFill というプロシージャを使いますが，塗り方 FillStyle が fsSurface と fsBorder の 2 種類あります。

1. `FloodFill(X,Y,` 面の色`,fsSurface)`
   点 (X,Y) と連結している “面の色” の点の集合を塗ります。

2. `FloodFill(X,Y,` 境界の色`,fsBorder)`
   点 (X,Y) を含み，“境界の色” で囲まれた領域を塗ります。言い換えると，(X,Y) と連結している “境界の色” 以外の色の点の集合を塗ります。

いずれにしても，三角形の内部の点 (X,Y) を指定しないといけません。簡単に計算できて，確実に内部にある点は**重心**です。

三角形 ABC の重心 G

$$\mathtt{G.X} = (\mathtt{A.X} + \mathtt{B.X} + \mathtt{C.X})/3$$
$$\mathtt{G.Y} = (\mathtt{A.Y} + \mathtt{B.Y} + \mathtt{C.Y})/3$$

[1] 理由はわかりませんが，左クリックではカーソルの形が変わりません。

6.3.1 関数 Juushin の定義

```
(******************** 一般のプロシージャ ********************)
function Juushin(A,B,C : TPoint) : TPoint;
         (* 三角形 ABC の重心 *)
var
  X,Y : Integer;
begin
  X := Round((A.X+B.X+C.X)/3);
  Y := Round((A.Y+B.Y+C.Y)/3);
  Juushin := Point(X,Y);
end; (* Juushin *)
```

6.3.2 Triangle を変更

```
procedure TFormMain.Triangle(A,B,C : TPoint);
          (* 三角形 ABC を描く *)
var
  TenG : TPoint;
begin
  with ImageMain.Canvas do
    begin
      MoveTo(A.X,A.Y);
      LineTo(B.X,B.Y);
      LineTo(C.X,C.Y);
      LineTo(A.X,A.Y);
      TenG := Juushin(A,B,C);
      Brush.Color := clRed;
      FloodFill(TenG.X,TenG.Y,Pixels[TenG.X,TenG.Y],fsSurface);
      //FloodFill(TenG.X,TenG.Y,Pen.Color,fsBorder);
    end;
end; (* Triangle *)
```

Pixels[X,Y] は点 (X,Y) の色を表します。

6.3.3 実行

FillStyle を 2 種類試してみましょう。ここでは，三角形の周がペンの色（黒）で内部が（白）一色なので，FillStyle をどちらにしても同じです。

6.4　外接円も描く

2 点 A, B を結ぶ直線上の点 P は次のように表されます。

直線上AB の点P の位置ベクトル

$$\overrightarrow{\boldsymbol{p}} = \frac{\alpha\overrightarrow{\boldsymbol{a}} + \beta\overrightarrow{\boldsymbol{b}}}{\alpha+\beta} \quad (\alpha:\beta = \mathrm{PB}:\mathrm{AP})$$

ただし，PB,AP は符号付の長さで，たとえば P → B が A → B と同じ向きのとき正で，逆向きのとき負とします。P が線分AB上にあるときは，α,βとも正で，P は AB の $\beta:\alpha$ の内分点になります。また，AB が軽くて丈夫な棒だとして，A に αグラムのおもり，B に βグラムのおもりを下げたとき，点Pを支点にするとABが水平に吊りあいます。この点PをABの荷重重心といいます。

同様に，3 点 A, B, C を含む平面上の点 P は次のように表されます。

平面ABC 上の点P の位置ベクトル

$$\overrightarrow{\boldsymbol{p}} = \frac{\alpha\overrightarrow{\boldsymbol{a}} + \beta\overrightarrow{\boldsymbol{b}} + \gamma\overrightarrow{\boldsymbol{c}}}{\alpha+\beta+\gamma} \quad (\alpha:\beta:\gamma = \triangle\mathrm{PBC}:\triangle\mathrm{APC}:\triangle\mathrm{ABP})$$

△PBC などは符号付の面積で，P → B → C が A → B → C と同じ回転方向（時計回りか反時計回りか）のとき正，逆回転のとき負とします。やはり，P は ABC の荷重重心になります。すなわち，A, B, C にそれぞれ α,β,γのおもりを下げたときに P を支点にするとつりあいます。

ふつうの重心は，$\alpha=\beta=\gamma$ のときの荷重重心と一致します。

重心の荷重

$$\alpha:\beta:\gamma \;=\; 1:1:1$$

外接円の中心，すなわち外心 O は，OA = OB = OC（外接円の半径） なので

$$\begin{aligned}
\alpha:\beta:\gamma &= \triangle\mathrm{OBC}:\triangle\mathrm{AOC}:\triangle\mathrm{ABO} \\
&= \sin\angle\,\mathrm{BOC}:\sin\angle\,\mathrm{COA}:\sin\angle\,\mathrm{AOB} \\
&= \sin 2A:\sin 2B:\sin 2C \\
&= \cdots\cdots \\
&= a^2(b^2+c^2-a^2):b^2(c^2+a^2-b^2):c^2(a^2+b^2-c^2)
\end{aligned}$$

となります。

外心の荷重

$$\begin{aligned}\alpha : \beta : \gamma &= \sin 2A : \sin 2B : \sin 2C \\ &= a^2(b^2+c^2-a^2) : b^2(c^2+a^2-b^2) : c^2(a^2+b^2-c^2)\end{aligned}$$

6.4.1 関数 Juushin 変更

Juushin を荷重重心を求める関数に変更します。

```
function Juushin(A,B,C : TPoint; Alpha,Beta,Gamma : Real) : TPoint;
         (* 三角形 ABC の荷重重心 *)
var
  X,Y : Integer;
begin
  X := Round((Alpha*A.X+Beta*B.X+Gamma*C.X)/(Alpha+Beta+Gamma));
  Y := Round((Alpha*A.Y+Beta*B.Y+Gamma*C.Y)/(Alpha+Beta+Gamma));
  Juushin := Point(X,Y);
end; (* Juushin *)
```

6.4.2 外接円の中心と半径を求めるのに必要な数学の関数

Juushin の定義の下に追加します。

```
end; (* Juushin *)

function Nagasa2Jou(A,B : TPoint) : Real;
         (* 線分 AB の長さの２乗 *)
begin
  Nagasa2Jou := Sqr(A.X-B.X)+Sqr(A.Y-B.Y);
end; (* Nagasa2Jou *)

function Nagasa(A,B : TPoint) : Real;
         (* 線分 AB の長さ *)
begin
  Nagasa := Sqrt(Sqr(A.X-B.X)+Sqr(A.Y-B.Y));
end; (* Nagasa *)

function Gaishin(A,B,C : TPoint) : TPoint;
         (* 三角形の外心 *)
var
  AB2,BC2,CA2 : Real;
begin
  AB2 := Nagasa2Jou(A,B);
  BC2 := Nagasa2Jou(B,C);
  CA2 := Nagasa2Jou(C,A);
  Gaishin := Juushin(A,B,C,BC2*(CA2+AB2-BC2),CA2*(AB2+BC2-CA2),AB2*(BC2+CA2-AB2));
end; (* Gaishin *)
```

6.4.3 Triangle を変更

外接円も描くようにする。

```
procedure TFormMain.Triangle(A,B,C : TPoint);
          (* 三角形 ABC を描く *)
          (* 外接円も描く     *)
var
  TenG : TPoint;
  TenO : TPoint;
  OA : Integer;
begin
  with ImageMain.Canvas do
    begin
      MoveTo(A.X,A.Y);
      LineTo(B.X,B.Y);
      LineTo(C.X,C.Y);
      LineTo(A.X,A.Y);
      TenG := Juushin(A,B,C,1,1,1);
      Brush.Color := clRed;
      //FloodFill(TenG.X,TenG.Y,Pixels[TenG.X,TenG.Y],fsSurface);
      FloodFill(TenG.X,TenG.Y,Pen.Color,fsBorder);

      TenO := Gaishin(A,B,C);
      OA := Round(Nagasa(TenO,A));
      Ellipse(TenO.X-OA,TenO.Y-OA,TenO.X+OA,TenO.Y+OA);
    end;
end; (* Triangle *)
```

6.4.4 実行

これでは三角形が見えません。

Ellipse は楕円を描きますが，その後で中をブラシで塗るのです。ですから，大きい外接円を先に描いてから，小さい三角形を描かないといけません。修正するついでに，色とブラシの形を違えてみましょう。

6.4.5 Triangle 修正

外接円を描く部分を上に移動して，ブラシの色と形の変更を追加する。

```
procedure TFormMain.Triangle(A,B,C : TPoint);
          (* 三角形 ABC を描く *)
          (* 外接円も描く     *)
var
  TenG : TPoint;
  TenO : TPoint;
  OA : Integer;
begin
  with ImageMain.Canvas do
    begin
      TenO := Gaishin(A,B,C);
      OA := Round(Nagasa(O,A));
      Brush.Color := clBlue;
```

```
      Brush.Style := bsFDiagonal;  // 斜線
      Ellipse(O.X-R,O.Y-R,O.X+R,O.Y+R);

      MoveTo(A.X,A.Y);
      LineTo(B.X,B.Y);
      LineTo(C.X,C.Y);
      LineTo(A.X,A.Y);
      TenG := Juushin(A,B,C,1,1,1);
      Brush.Color := clRed;
      Brush.Style := bsSolid;
      //FloodFill(TenG.X,TenG.Y,Pixels[TenG.X,TenG.Y],fsSurface);
      FloodFill(TenG.X,TenG.Y,Pen.Color,fsBorder);
    end;
end; (* Triangle *)
```

6.4.6　実行

外接円と三角形が見えます。

外接円の内部が一色でないので，FloodFill の FillStyle を変えると違いがわかります。FillStyle を変えたり，外接円の Brush.Style をいろいろ変えて実行してみてください。

注意　点Aが直線BC上にある（三角形にならない）と，外心の計算で $\alpha+\beta+\gamma=0$ で割り算することになり実行時エラーとなってしまいます。Aが直線BCを横切るときは，マウスを素早く動かして，BC上で MouseMove イベントが生じないようにしてください。エラーになった場合は，[実行｜プログラムの終了] をして強制終了してください。

6.4.7　ClearGraph 変更

Triangle の中で最後に設定したブラシスタイルが，ClearGraph でも適用されるので，三角形を bsSolid 以外のスタイルで描くと，画面消去が正常になされません。ClearGraph の中で bsSolid に設定しなおします。

```
\textit{  with ImageMain.Canvas do
    begin
      Brush.Color := Color;}
      Brush.Style := bsSolid;
\textit{      FillRect(ClipRect);
    end;}
```

6.5　内接円も描く

内接円の中心，すなわち内心 I は，Iから各辺に下ろした垂線の長さが等しい（内接円の半径）ので

$$\alpha:\beta:\gamma \quad = \quad \triangle\mathrm{IBC}:\triangle\mathrm{AIC}:\triangle\mathrm{ABI} \quad = \quad a:b:c$$

内心の荷重

$$\alpha : \beta : \gamma \quad = \quad a : b : c$$

内接円の半径の求め方はいろいろありますが，ここでは垂線の長さを用いることにします。
一般に三角形ABC の頂点Cから対辺ABに下ろした垂線の足をHとすると，

$$\begin{aligned} \mathrm{HB} : \mathrm{AH} &= a\cos B : b\cos A \\ &= 2ca\cos B : 2bc\cos A \\ &= c^2 + a^2 - b^2 : b^2 + c^2 - a^2 \end{aligned}$$

垂線の足の荷重

$$\alpha : \beta = c^2 + a^2 - b^2 : b^2 + c^2 - a^2$$

6.5.1 Triangle を変更

内接円も描くようにします。

```
procedure TFormMain.Triangle(A,B,C : TPoint);
          (* 三角形 ABC を描く *)
          (* 外接円も描く     *)
          (* 内接円も描く     *)
var
  TenG : TPoint;
  TenO : TPoint;
  TenI : TPoint;
  OA : Integer;
  IH : Integer;
begin
  with ImageMain.Canvas do
    begin
      TenO := Gaishin(A,B,C);
      OA := Round(Nagasa(TenO,A));
      Brush.Color := clLime;
      Brush.Style := bsBDiagonal;
      Ellipse(TenO.X-OA,TenO.Y-OA,TenO.X+OA,TenO.Y+OA);

      MoveTo(A.X,A.Y);
      LineTo(B.X,B.Y);
      LineTo(C.X,C.Y);
      LineTo(A.X,A.Y);
      TenG := Juushin(A,B,C,1,1,1);
      Brush.Color := clRed;
      Brush.Style := bsSolid;
      //FloodFill(TenG.X,TenG.Y,Pixels[TenG.X,TenG.Y],fsSurface);
      FloodFill(TenG.X,TenG.Y,Pen.Color,fsBorder);
```

```
      TenI := Naishin(A,B,C);
      IH := Round(Nagasa(TenI,SuisenNoAshi(A,B,TenI)));
      Brush.Color := clBlue;
      Brush.Style := bsFDiagonal;
      Ellipse(TenI.X-IH,TenI.Y-IH,TenI.X+IH,TenI.Y+IH);
    end;
end; (* Triangle *)
```

6.5.2　内接円の中心と半径を求めるための数学の関数

未完成部分を完成させてください。

```
function Naishin(A,B,C : TPoint) : TPoint;
         (* 三角形 ABC の内心 *)
var
  AB,BC,CA : Real;
begin
  ４行



end; (* Naishin *)

function Bunten(A,B : TPoint; Alpha,Beta : Real) : TPoint;
         (* AB の β:α の分点 *)
var
  X,Y : Integer;
begin
  ３行


end; (* Bunten *)

function SuisenNoAshi(A,B,C : TPoint) : TPoint;
         (* C から AB に下ろした垂線の足 *)
var
  AB2,BC2,CA2 : Real;
begin
  ４行



end; SuisenNoAshi
```

6.5.3　実行

最初の三角形は正三角形なので上の 3 つの関数が間違っていても内接円が正しく描かれることがあります。頂点 A を動かしても内接円が正しく描かれることを確認したください。